1993-04

New Software-Quality Metrics Methodology Standard fills measurement need

Schneidewind, Norman F.

Editor: Norman F. Schneidewind, Naval Postgraduate School, Code AS/Ss, Monterey, CA 93943, fax (408) 656-3407, e-mail 0442p@vm1.cc.nps.navy.mil

New Software-Quality Metrics Methodology Standard fills measurement need

Norman F. Schneidewind, Naval Postgraduate School and 1061 Working Group Chair

Last December, the IEEE Standards Board approved the Standard for a Software-Quality Metrics Methodology, 1061, the first IEEE-issued standard that deals with quality metrics.

As a process standard, 1061 does not mandate specific metrics for use. One reason for this approach is that industry needs a methodology for implementing a metrics plan that is independent of particular metrics that might be used.

Selling this approach to the metrics community has not been easy because a significant part of that community feels compelled to "measure something" and to "put measurement into practice." The 1061 Working Group subscribes to the objective of putting "measurement into practice" as long as there is a *plan* to support it; 1061 provides the plan.

Philosophy behind 1061. The philosophy of 1061 is that an organization can use whichever metrics it deems most appropriate for its applications as long as the methodology is followed and the metrics are validated (see below).

Another reason for this approach is that the 1061 Working Group did not reach a consensus about which metrics should be mandated for use (the provisions of a standard are mandatory, not optional). Consistent with this approach was the working group's charter, as provided in the IEEE Standards Board approval of the project authorization request, which called for development of a standard methodology.

However, despite its reservations about endorsing specific metrics, the working group provided definitions and descriptions of several popular metrics in appendixes (see below), which are included in the 1061 document but are not part of the standard.

Many empirical studies conducted to validate various metrics have not yielded consistent results. What is needed is a theoretical underpinning for the validation and application of metrics, coupled with large-scale experiments on the statistical relationship between quality factors such as time-to-failure (an example of a product characteristic of interest to customers) and early indicators of quality such as complexity (a product characteristic of interest to developers).

Quality factors and indicators are both metrics. However, one point that 1061 makes is that often the metric (that is, quality factor) of greatest interest to the customer cannot be measured in an early phase of the software-development process; instead, a metric such as complexity can be used as an indicator of quality during development.

To achieve high software quality in a system, the software's attributes must be clearly defined; otherwise, assessment of quality is left to intuition.

Theories of measurement and validation have emerged to provide a rationalization for metric selection and application.[1-4] This development, coupled with the identification of composite metrics that show a significant relationship with quality factors,[5] led the working group to believe that a set of quality metrics, anchored in both theory and practice, could be published as a standard in the next several years.

Scope of the standard. This standard provides a methodology for establishing quality requirements and identifying, implementing, analyzing, and validating software-quality metrics. This methodology applies to all software at all phases of any software life-cycle structure. As stated above, the 1061 standard does not prescribe specific metrics.

Standard's audience. This standard is intended for those associated with the acquisition, development, use, support, maintenance, or audit of software, and is aimed particularly at those measuring or assessing the quality of software.

The standard can be used by

- acquisition or project managers, to identify and prioritize the quality requirements for a system;
- system developers, to identify specific traits that should be built into the software to meet the quality requirements;
- quality assurance organizations and system developers, to evaluate whether quality requirements are being met;
- system maintainers, to assist in change management during product evolution; and
- users, to assist in specifying the requirements of the requested system.

Why software-quality metrics? Software quality is the degree to which software possesses a desired combination of attributes. To achieve high software quality in a system, this combination of attributes must be clearly defined; otherwise, assessment of quality is left to intuition.

For this standard, defining software quality for a system is equivalent to defining a list of software-quality attributes required for that system. An appropriate set of metrics must be identified to measure the software-quality attributes.

The purpose of software-quality metrics is to make assessments throughout the software life cycle as

to whether the software-quality requirements are being met. Using metrics reduces subjectivity in software-quality assessment by providing a quantitative basis for making decisions. However, such usage does not eliminate the need for human judgment in software evaluations.

Organizations and projects that use these metrics can expect to enjoy the beneficial effect of more visible software quality.

Methodology of metrics. The software-quality metrics methodology is a systematic approach to establishing quality requirements and identifying, implementing, analyzing, and validating process and product software-quality metrics for a software system. It spans the entire software life cycle and comprises five steps. These steps must be applied iteratively because insights gained from applying a step can show the need for further evaluation of the results of prior steps. The steps are:

(1) *Establish software-quality requirements.* A list of quality factors is selected, prioritized, and quantified at the outset of system development or system change. These requirements are to be used to guide and control development of the system and, on delivery of the system, to assess whether the system meets the quality requirements specified in the contract.

(2) *Identify software-quality metrics.* The software-quality metrics framework is applied in selecting relevant metrics.

(3) *Implement the software-quality metrics.* Tools are procured or developed, data is collected, and metrics are applied at each phase of the software life cycle.

(4) *Analyze the software-quality metrics results.* These results are analyzed and reported to help control development and assess the final product.

(5) *Validate the software-quality metrics.* Predictive metrics results are compared with quality factor results to determine whether the predictive metrics accurately measure their associated factors.

Why metrics validation? The purpose of metrics validation is to identify product and process metrics that can predict specified quality factors, the quantitative representations of quality requirements. If metrics are to be useful, they must indicate accurately whether quality requirements have been achieved or are likely to be achieved in the future.

When it is possible to measure factor values at the desired point in the life cycle, these quality factors are used to evaluate software quality. At some points in the life cycle, certain quality factors (for example, reliability) are not available; they are obtained after delivery or late in the project. In these cases, other metrics are used early in a project to predict quality factors.

The history of applying metrics indicates that metrics were seldom validated (that is, it was not demonstrated through statistical analysis that the metrics actually measured software characteristics as they purported to).

However, validating metrics before they are used to evaluate software quality is important. Otherwise, metrics might be misapplied (that is, metrics might be used that have little or no relationship to the desired quality characteristics).

Quality factors may be affected by multiple variables. A particular metric, therefore, might not sufficiently represent any single factor because it ignores other variables.

Validity criteria. To be considered valid, a metric must demonstrate a high degree of association with the quality factors it represents. This is equivalent to accurately portraying the quality condition(s) of a product or process. A metric may be valid with respect to certain validity criteria and invalid with respect to other criteria. The following criteria are used:

Correlation. The variation in the quality factor values for a product or process explained by the variation in the corresponding metric values must exceed a specified threshold.

Tracking. A change (increase or decrease) in a quality factor value for a product or process must be accompanied by a corresponding change (increase or decrease, as appropriate) in a metric value.

Consistency. If quality factor values are rank-ordered for products or processes, the corresponding metric values must have the same ordering.

Predictability. If a metric is used to predict a quality factor for a product or process, it must predict within a specified accuracy.

Discriminative power. A metric must be able to discriminate between high- and low-quality products or processes.

Reliability. A metric must demonstrate the above correlation, tracking, consistency, predictability, and discriminative power properties for a specified percentage of the applications of the metric.

Appendixes. Several appendixes in the standard provide definitions and examples of commonly used quality factors and metrics. Unique features of this standard are appendixes that contain numerical results from using popular metrics and comprehensive case studies — one in the mission-critical area (aerospace and defense software development) and another in the commercial arena (off-the-shelf commercial software) — that illustrate the application of the methodology in a step-by-step fashion.

Appendix A contains examples of factors, subfactors, and metrics, and the relationships among them; Appendix B, sample metrics descriptions and computations; Appendix C, examples of methodology usage (for instance, mission-critical or commercial environment); and Appendix D, annotated bibliography, references, and standards.

To go further. Copies of Standard 1061 can be obtained by contacting the IEEE Standards Office, 445 Hoes Lane, PO Box 1331, Piscataway, NJ 08855-1331, phone (908) 562-3800. Please do not request copies from the author.

Acknowledgments

This standard was developed by a broad-based committee of representatives from industry, government, and academe. I thank the members of the working and balloting groups and the many individuals who contributed comments toward development of the standard.

References

1. H. Zuse, *Software Complexity: Measures and Methods*, Walter de Gruyter, 1991.
2. N.E. Fenton, *Software Metrics: A Rigorous Approach*, Chapman and Hall, 1991.
3. N.F. Schneidewind, "Methodology for Validating Software Metrics," *IEEE Trans. Software Eng.*, Vol. 18, No. 5, May 1992, pp. 410-422.
4. N.F. Schneidewind, "Minimizing Risks in Applying Metrics on Multiple Projects," *Proc. Third Int'l Symp. Software Reliability Eng.*, IEEE CS Press, Los Alamitos, Calif., Order No. 2975, 1992, pp. 173-182.
5. J.C. Munson and T.M. Khoshgoftarr, "The Detection of Fault-Prone Programs," *IEEE Trans. Software Eng.*, Vol. 18, No. 5, May 1992, pp. 423-433.